Nesta aula estudamos um dos principais conceitos da orientação a objetos: o que são classes, de que forma são utilizadas e como funciona a herança de classe.

Recapitulando o que vimos, assinale as alternativas corretas:

---

A herança de classe é importante para um melhor reaproveitamento de código, uma vez que permite a criação de novas funcionalidades com base em um modelo; além disso, faz com que os objetos e as regras de negócio criadas pelo sistema façam sentido e de fácil abstração.

Alternativa correta! Com a herança de classe, podemos organizar o sistema de forma que suas entidades façam sentido entre si, e reaproveitamos melhor o código. Por exemplo: seres humanos, gatos e pássaros são animais e poderiam "herdar" propriedades de uma superclasse `Animal`, com propriedades e métodos em comum. Assim pode acontecer com carros, comidas, usuários de um sistema, etc.

As classes não são a forma nativa do JavaScript trabalhar com orientação a objetos e foram desenvolvidas sobre o modelo de protótipo.

Alternativa correta! Como fizemos durante a aula, é possível verificar isso através do método `isPrototypeOf()`, como no exemplo:

```
NomeDaClasse.prototype.isPrototypeOf(subclasse);
```

COPIAR CÓDIGO

O `constructor()` é uma função especial que recebe, via parâmetros, as propriedades que um objeto precisa ter ao ser instanciado a partir de uma classe; também é através do construtor que uma classe herda métodos e propriedades da superclasse através da função `super()`. Porém, dependendo da necessidade do projeto, uma classe pode não ter um construtor, apenas métodos.

Alternativa correta! Apesar de ser necessário quando uma classe precisa receber propriedades no momento em que é instanciada e também para que a herança de classe funcione, o construtor não é obrigatório; uma classe pode ter somente métodos ou não receber nenhuma propriedade no momento em que é instanciada.

**D**

Após a implementação das classes, as funções construtoras estão sendo descontinuadas.

Apesar da grande adesão às classes, especialmente por devs que já tinham contato anterior com esta sintaxe, as formas anteriores de se trabalhar com orientação a objetos utilizando funções não foram descontinuadas e são utilizadas.